Fault Tolerant System Design

Dr. Axel Krings
JEB 320
208 885-4078
krings@uidaho.edu
http://www.cs.uidaho.edu/~krings

What is Fault Tolerance ?

Why use Fault Tolerance?

It is Written:
"To err is human, but to ***really*** foul up takes a computer"

- Computers are used where system failure would be catastrophic in terms of money, human lives, or ecosystem.
- Applications: Process Control, Patient Monitoring Systems, Missile guidance & Control, Air Traffic Control, Fly-by-Wire Aircraft, Transaction Processing, Stock Market

Fault-Tolerant System Design

- Different flavors, e.g.
  - General Fault Tolerance
  - Design for Testability
  - FT for safety critical applications
  - Hardware Fault Tolerance
  - Software Fault Tolerance
  - Supersets:
    - » Survivability
    - » Resilience
    - » ...

Introduction

- Designing Safety-Critical Computer Systems
  - the discussion below is directly drawn from the same-called article by William R. Dunn, IEEE Computer, Vol. 36 , Issue 11 (November 2003), Pages: 40-46.
- More and more computers are used to control safety-critical applications
  - fly-by-wire, hospital life-support systems, manufacturing robots etc.
  - coming up: steer-by-wire automotive systems, automated air- and surface-traffic control, powered prosthetics, smart Grid, etc.

Introduction

- Concern: can these systems fail and cause harm?
  - example: Therac 25 therapeutic computer system accidents
- Concern: proposed system concepts and architectures
  - have been found to be impractical for safety critical real-life engineering applications
  - fail in practice for three primary reasons:
    - originators or users
      - have incomplete understanding of what makes a system safe
      - fail to consider the larger system into which the system in integrated
      - ignoring single point of failure

Introduction

- Therac-25 was a radiation therapy machine produced by Atomic Energy of Canada Limited (AECL) and CGR of France after the Therac-6 and Therac-20 units. It was involved with at least six known accidents between 1985 and 1987, in which patients were given massive overdoses of radiation, which were in some cases on the order of hundreds of grays. At least five patients died of the overdoses. These accidents highlighted the dangers of software control of safety-critical systems, and they have become a standard case study in health informatics.
- source: wikipedia.org

Introduction

- Defining "Safe"
  - we often think "safe" w.r.t. driving a car, flying etc.
    - e.g. "is it safe to drive?"
    - one thinks of a *mishap*
  - Mishap
    - MIL-STD-882D definition: "An unplanned event or series of events resulting in death, injury, occupational illness, damage to or loss of equipment or property, or damage to the environment."
  - Mishap Risk
    - MIL-STD-882D definition: "An expression of the impact and possibility of a mishap in terms of potential mishap severity and probability of occurrence."
    - Example: airline crash vs. fender-bender: less likely, but higher impact
    - What is the important message here:
      - **Systems are never absolutely safe => thus reduce risk...**

Introduction

- Acceptable Mishap Risk
  - public establishes acceptable risk for a given mishap
  - willingness to tolerate mishap as long as it occurs infrequently
  - typical fail rates: 10^{-2} to 10^{-10} per hour
  - how do designers decide on what constitutes an acceptable risk?
    - they don't!
    - they rely on standards such as
      - MIL-STD-882D
      - IEC 61508, Functional safety of electrical/electronic/ programmable electronic safety-related systems.

Introduction

- Computer System
  - Application
    - » physical entity the system controls/monitors, e.g. plant, process
  - Sensor
    - » converts application's measured properties to appropriate computer input signals, e.g. accelerometer, transducer
  - Effector
    - » converts electrical signal from computer's output to a corresponding physical action that controls function, e.g. motor, valve, break, pump.
  - Operator
    - » human(s) who monitor and activate the computer system in real-time, e.g. pilot, plant operator, medical technician
  - Computer
    - » hardware and software that use sensors and effectors to control the application in real-time, e.g. single board controller, programmable logic controller, flight computers, systems on a chip.

Introduction

- Hazard Analysis
  - Hazard
    - » MIL-STD-882D definition: "Any real or potential condition that can cause injury, illness, or death to personnel; damage to or loss of a system, equipment or property; or damage to the environment."
  - examples: loss of flight control, nuclear core cooling, presence of toxic materials or natural gas

Introduction

- System design
  - identify hazards of application components
  - next, determine how operator, sensor, computer and effectors can fail and cause mishaps
    - » use failure-modes analysis to discover all possible failure sources in each component, i.e. operator, sensor, computer and effector
    - » includes random hardware failure, manufacturing defects, program faults, environmental stresses, design errors, maintenance mistakes
  - now the design can begin

Introduction

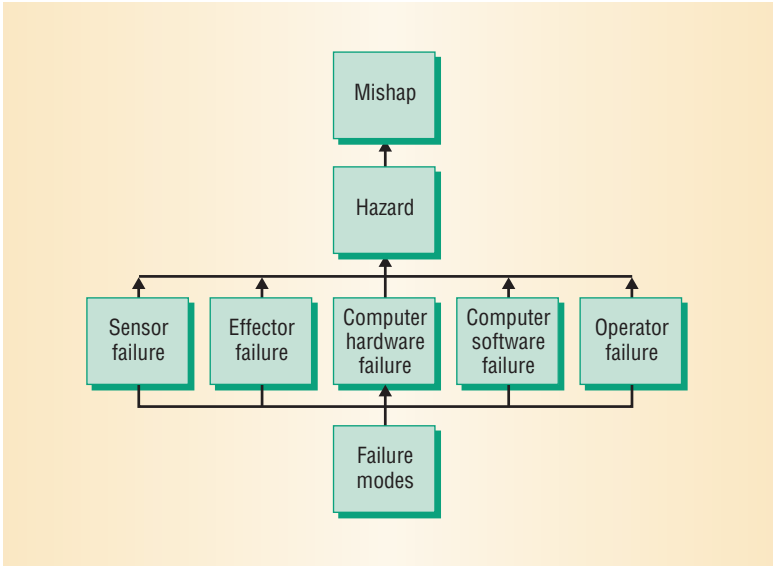

Figure 1. Mishap causes. System designers identify the application's attendant hazards to determine how system-component failures can result in mishaps.

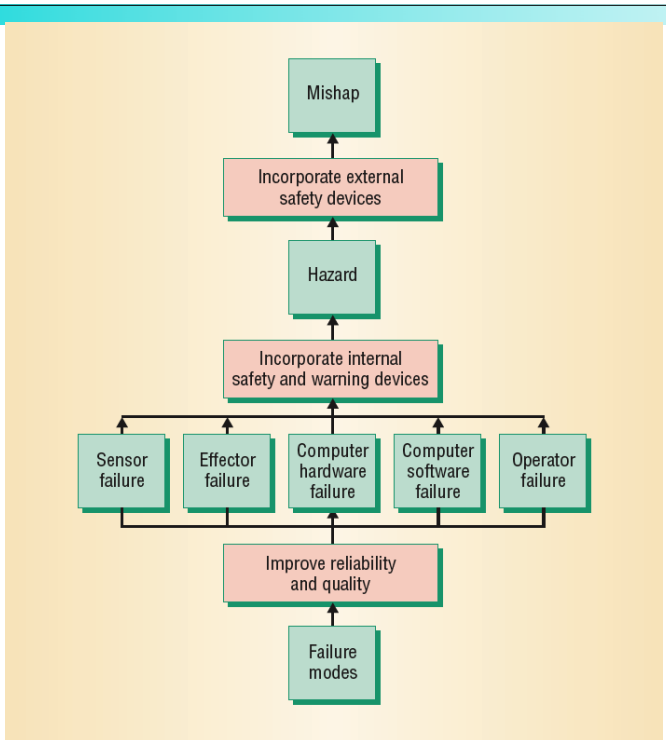

Figure 2. Risk mitigation measures. Designers can modify a system to reduce its inherent risk by improving component reliability and quality and by incorporating internal or external safety and warning devices.

Introduction: Example

- Example: computer system used for electrically heating water
  - Application
    - » steel tank containing water
  - Effector
    - » computer-controlled electric heating elements
  - Sensor
    - » temperature sensor measures water temp and transmits to computer
  - Computer
    - » software in the computer maintains water temp at 120F by controlling heating element
      - ■ ON if water temperature is below target
      - ■ OFF otherwise

Introduction Example

- Example cont.
  - Hazard
    - e.g. water could overheat
  - Mishap
    - e.g. overheated water could cause tank to explode
    - e.g. person opens faucet and gets scald by overheated water or steam
  - Failures that could create this hazard
    - temperature sensor malfunction signaling “low temperature”
    - heater unit may fail and remain on permanently
    - computer interface hardware might fail permanently signaling an “ON” state to the heater
    - computer software fault, possibly in unrelated routine, might change the set point to 320F
    - operator might program an incorrect set point

Introduction

- Failures that could create this hazard, (cont.)
  - maintenance error, e.g. repair person installs wrong temperature sensor.
  - environmental condition, e.g. overly warm application location causes chips to fail
  - design failure that results in using the wrong sensor for the selected operating temperature.
- This water heating system (as it stands) has unacceptable risk of mishap!

Introduction

- Mishap Risk Mitigation
  - Options:
    - 1) improve component reliability and quality
      - seeks to lower probability of component failure
      - which in turn reduces probability of mishap
    - 2) incorporate internal safely and warning devices
      - e.g. thermocouple device turns off gas to home heater when pilot goes out
    - 3) incorporate external safety devices
      - range from simple physical containment to computer-based safety-instrumented systems
  - Designers should apply all of these options
    - ensure distributed, **non-single-point-of-failure** implementation

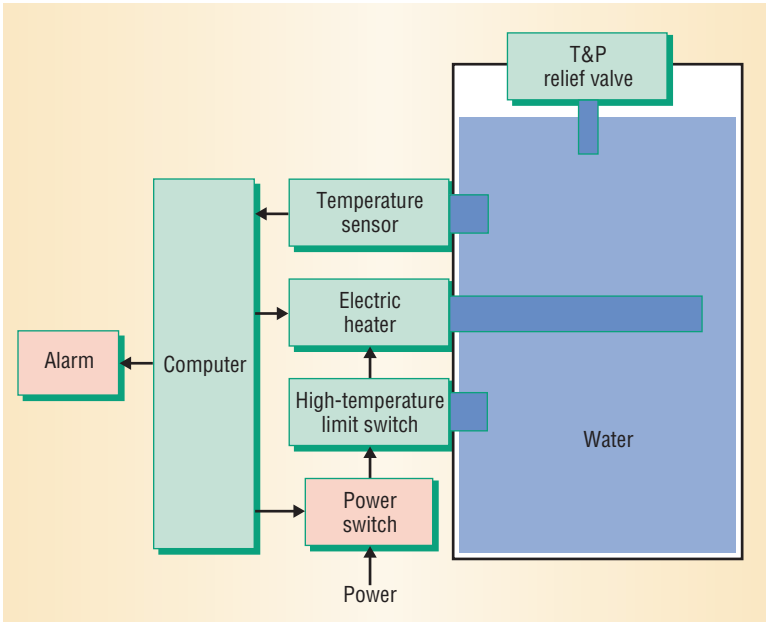

Figure 3. Applying risk-mitigation measures. The addition of safety devices such as a high-temperature limit switch and a temperature-and-pressure (T&P) relief valve has reduced the computer-controlled water heating system's operational risk.

Introduction

- Additional Safety Devices

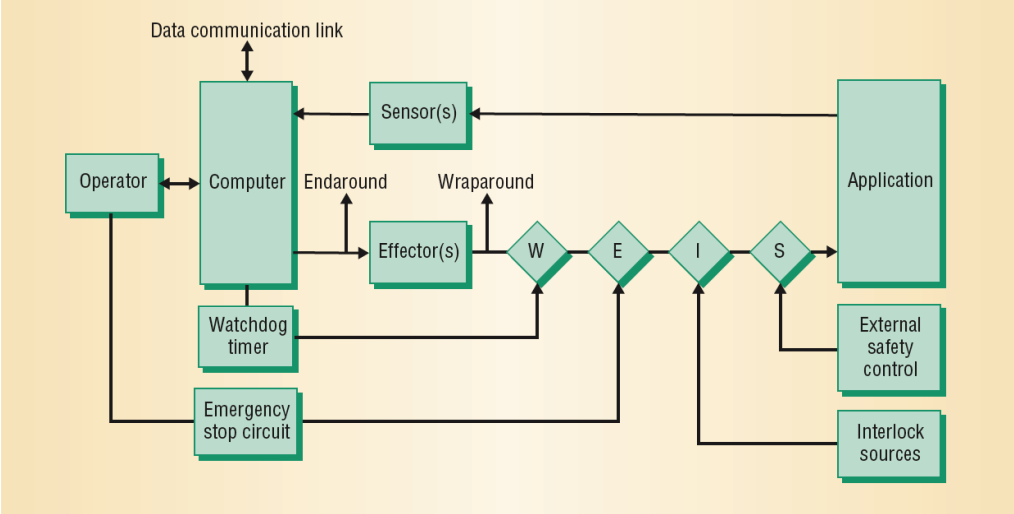

Figure 4. Risk mitigation methods. Designers have added several risk-mitigation devices to this system, including a watchdog timer, emergency stop circuit, and interlocks that inhibit effector actions unless specific external conditions are satisfied.

Introduction

- Fail-Operate Systems
  - Fail-Safe System
    - » after failure is detected, systems enters a safe state, by modifying effector outputs, e.g. shut system down.
  - Fail-Operate System
    - » many computer systems cannot just be shut down
    - » e.g. fly-by-wire aircraft control system
    - » system must continue safe operation even after one or more components have failed
    - » tolerating faults is the goal of fault-tolerant system design
    - » strongly relies on the principle of redundancy

Introduction

- Fail-Operate System
  - » principle of redundancy is simple in concept, but hard to implement
  - » all critical system components must be replicated
    - i.e. computers, sensors, effectors, operators, power source, interconnect.
    - ... not to mention the issue of homogeneous vs inhomogeneous redundancy (identical vs dissimilar)
  - » redundancy management needs to be incorporated into hardware, software, operator components
    - detect failure
    - isolate failed component
    - reconfigure components
    - we will address reconfiguration and masking extensively later in the course
  - » system cost and complexity increase fast

Introduction

- Evaluating Safety-Critical Computer Systems
  - Failure Modes and Effects Analysis (FMEA)
    - » for each component consider how it can fail, then determine the effects each failure has on the system
    - » goal is to identify single point of failure
  - Fault-Tree Analysis (FTA)
    - » identify mishap and identify all components that can cause a mishap and all the safety devices that can mitigate it.
  - Risk Analysis (RA)
    - » quantitative measure yielding numerical probabilities of mishap
    - » need failure probabilities of components

Introduction

- Reliability Modeling
  - » considering all components, redundant and non-redundant, determine the probability that the system will (reliability) or will not (unreliability) operate correctly (one hour typical)

- Design Strategy
  - » use fault tree to evaluate overall probability of failure
  - » can consult probabilities of fault tree to identify where to apply mitigation
  - » need to re-design sections that contribute heavily to unreliability
  - » continue this process until desired reliability is achieved

Finding a Compromise

How much fault-tolerance is needed for a system or application?

High cost vs. customer dissatisfaction/loss of market shares

Systems operate just below the threshold of pain

Top Five Challenges

- Ram Chillarege (1995) writes:

  The top 5 challenges, which ultimately drive the exploitation of fault-tolerant technology are:

  1) Shipping a product on schedule
  2) Reducing Unavailability
  3) Non-disruptive Change Management
  4) Human Fault Tolerance
  5) Distributed Systems

  Article source: Lecture Notes In Computer Science; Vol. 774, 1999
  - the points made in the article still hold

Shipping Product on Schedule

- extreme pressure to reduce product cycle
- competitive market
  - introduce products faster
- FT adds cost in Hardware, Design, Verification
  - increase development cycle
- compressed schedule can result in greater # of errors
  - errors escape into field

Reducing Unavailability

* Outage and their Impacts:
  -- software & procedural issues (operator errors)
  -- hardware & environmental problems
* years ago: Hardware problems dominant
* Improvements in manufacturing & technology
* Improvements in software not significant
  -- software problems now dominate outages
* Software Bugs:
  total failure < 10%
  partial failure 20% - 40% (requires some response)
  rest: Annoyance, update later, update via maintenance

Reducing Unavailability cont.

* Down-Time (largest outage part)

- upgrades - maintenance - reconfiguration	planned outage
- act of technology/nature - commonly the target of FT design	unscheduled outage

* Some commercial applications
  - 24 x 7 operations
  - reduce outage from all sources

Non-Disruptive Change Management

* Maintenance on Software
  - most software is not designed to be maintained non-disruptive
* One Solution: hot standby
* The Problem of First Failure Data Capture (FFDC)
  - trap, trace, log adequate information
  - FFDC mostly poor
  - error propagation makes it harder to find root cause of problem
  - problems in re-creating

Human Fault Tolerance

* Human Comprehension of task = "non-defect oriented problem"
  - no code change required

* Design System to tolerate human error

Now consider Distributed Systems

We need to start "all over again"

Fault-Tolerance & Ultra Reliable Systems

Example: Fly-by-Wire, i.e. Airbus 320

- computer controls all actuators
- no control rods, cables in the middle
- 5 central flight control computers
- different systems used
  Thomson CSF => 68010
  SFENA => 80186
- software for both hardware written
  by different software houses
- all error checking & debugging performed separately
- computer allows pilot to fly craft up to certain limits
  - beyond: computer takes over

Airbus A320/A330/A340 Electrical flight Controls: A Family of Fault-Tolerant systems,
D. Briere, and P. Traverse, FTCS-23, pp.616-623, 1993.

Fault-Tolerance & Ultra Reliable Systems

* Many aircraft use active control
  F16
  forward swept wing X-29
  could not fly without computers
  moving control surfaces
* Burden of proof that fly-by-wire system is safe for civil flight has shifted to training environments and simulation.